关闭阅读模式

Spring Boot干货系列：（二）配置文件解析

2017-02-28

Spring Boot干货系列 Spring Boot

前言

上一篇介绍了Spring Boot的入门，知道了Spring Boot使用“习惯优于配置”（项目中存在大量的配置，此外还内置了一个习惯性的配置，让你无需手动进行配置）的理念让你的项目快速运行起来。所以，我们要想把Spring Boot玩的溜，就要懂得如何开启各个功能模块的默认配置，这就需要了解Spring Boot的配置文件application.properties。

正文

Spring Boot使用了一个全局的配置文件application.properties，放在src/main/resources目录下或者类路径的/config下。Sping Boot的全局配置文件的作用是对一些默认配置的配置值进行修改。

接下来，让我们一起来解开配置文件的面纱。

> 注:如果你工程没有这个application.properties，那就在src/main/java/resources目录下新建一个。

自定义属性

application.properties提供自定义属性的支持，这样我们就可以把一些常量配置在这里：

```
1 com.dudu.name="嘟嘟MD"
2 com.dudu.want="祝大家鸡年大吉吧"
```

然后直接在要使用的地方通过注解@Value(value=”${config.name}”)就可以绑定到你想要的属性上面

```
@RestController
public class UserController {

    @Value("${com.dudu.name}")
    private  String name;
    @Value("${com.dudu.want}")
    private  String want;

    @RequestMapping("/")
    public String hexo(){
        return name+","+want;
    }
}
```

我们启动工程输入http://localhost:8080 就可以看到打印了”嘟嘟MD祝大家鸡年大吉吧”。

有时候属性太多了，一个个绑定到属性字段上太累，官方提倡绑定一个对象的bean，这里我们建一个ConfigBean.java类，顶部需要使用注解@ConfigurationProperties(prefix = “com.dudu”)来指明使用哪个

```
@ConfigurationProperties(prefix = "com.dudu")
public class ConfigBean {
    private String name;
    private String want;

    // 省略getter和setter
}
```

这里配置完还需要在spring Boot入口类加上@EnableConfigurationProperties并指明要加载哪个bean，如果不写ConfigBean.class，在bean类那边添加

```
@SpringBootApplication
@EnableConfigurationProperties({ConfigBean.class})
public class Chapter2Application {

    public static void main(String[] args) {
        SpringApplication.run(Chapter2Application.class, args);
    }
}
```

最后在Controller中引入ConfigBean使用即可，如下：

```
@RestController
public class UserController {
    @Autowired
    ConfigBean configBean;

```

```
    public String hexo(){
        return configBean.getName()+configBean.getWant();
    }
}
```

参数间引用

在application.properties中的各个参数之间也可以直接引用来使用，就像下面的设置：

```
com.dudu.name="嘟嘟MD"
com.dudu.want="祝大家鸡年大吉吧"
com.dudu.yearhope=${com.dudu.name}在此${com.dudu.want}
```

这样我们就可以只是用yearhope这个属性就好

使用自定义配置文件

有时候我们不希望把所有配置都放在application.properties里面，这时候我们可以另外定义一个，这里我明取名为test.properties,路径跟也放在src/main/resources下面。

```
com.md.name="哟西~"
com.md.want="祝大家鸡年,大吉吧"
```

我们新建一个bean类,如下：

```
@Configuration
@ConfigurationProperties(prefix = "com.md")
@PropertySource("classpath:test.properties")
public class ConfigTestBean {
    private String name;
    private String want;
    // 省略getter和setter
}
```

这里要注意哦，有一个问题，如果你使用的是1.5以前的版本，那么可以通过locations指定properties文件的位置，这样：

```
@ConfigurationProperties(prefix = "config2",locations="classpath:test.properties")
```

但是1.5版本后就没有这个属性了，找了半天发现添加@Configuration和@PropertySource(“classpath:test.properties”)后才可以读取。

随机值配置

配置文件中${random} 可以用来生成各种不同类型的随机值，从而简化了代码生成的麻烦，例如 生成 int 值、long 值或者 string 字符串。

```
dudu.secret=${random.value}
dudu.number=${random.int}
```

```
dudu.uuid=${random.uuid}
dudu.number.less.than.ten=${random.int(10)}
dudu.number.in.range=${random.int[1024,65536]}
```

—— 外部配置-命令行参数配置 ——

Spring Boot是基于jar包运行的，打成jar包的程序可以直接通过下面命令运行：

```
java -jar xx.jar
```

可以以下命令修改tomcat端口号：

```
java -jar xx.jar --server.port=9090
```

可以看出，命令行中连续的两个减号 `--` 就是对 `application.properties` 中的属性值进行赋值的标识。
所以 `java -jar xx.jar --server.port=9090` 等价于在 `application.properties` 中添加属性 `server.port=9090` 。
如果你怕命令行有风险，可以使用SpringApplication.setAddCommandLineProperties(false)禁用它。

实际上，Spring Boot应用程序有多种设置途径，Spring Boot能从多重属性源获得属性，包括如下几种：

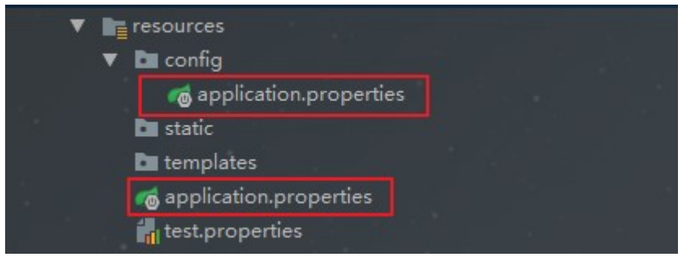
- 根目录下的开发工具全局设置属性(当开发工具激活时为 `~/.spring-boot-devtools.properties`)。
- 测试中的@TestPropertySource注解。
- 测试中的@SpringBootTest#properties注解特性。
- 命令行参数
- `SPRING_APPLICATION_JSON` 中的属性(环境变量或系统属性中的内联JSON嵌入)。
- `ServletConfig` 初始化参数。
- `ServletContext` 初始化参数。
- java:comp/env里的JNDI属性
- JVM系统属性
- 操作系统环境变量
- 随机生成的带random.* 前缀的属性（在设置其他属性时，可以应用他们，比如${random.long}）
- 应用程序以外的application.properties或者appliaction.yml文件
- 打包在应用程序内的application.properties或者appliaction.yml文件
- 通过@PropertySource标注的属性源
- 默认属性(通过 `SpringApplication.setDefaultProperties` 指定).

这里列表按组优先级排序，也就是说，任何在高优先级属性源里设置的属性都会覆盖低优先级的相同属性，列如我们上面提到的命令行属性就覆盖了application.properties的属性。

—— 配置文件的优先级 ——

- 外置，在相对于应用程序运行目录的/congfig子目录里。
- 外置，在应用程序运行的目录里
- 内置，在config包内
- 内置，在Classpath根目录

同样，这个列表按照优先级排序，也就是说，src/main/resources/config下application.properties覆盖src/main/resources下application.properties中相同的属性，如图：



此外，如果你在相同优先级位置同时有application.properties和application.yml，那么application.properties里的属性里面的属性就会覆盖application.yml。

—— Profile-多环境配置 ——

当应用程序需要部署到不同运行环境时，一些配置细节通常会有所不同，最简单的比如日志，生产日志会将日志级别设置为WARN或更高级别，并将日志写入日志文件，而开发的时候需要日志级别为DEBUG，日志输出到控制台即可。

如果按照以前的做法，就是每次发布的时候替换掉配置文件，这样太麻烦了，Spring Boot的Profile就给我们提供了解决方案，命令带上参数就搞定。

这里我们来模拟一下，只是简单的修改端口来测试
在Spring Boot中多环境配置文件名需要满足`application-{profile}.properties`的格式，其中`{profile}`对应你的环境标识，比如：

- application-dev.properties：开发环境
- application-prod.properties：生产环境

想要使用对应的环境，只需要在application.properties中使用spring.profiles.active属性来设置，值对应上面提到的{profile}，这里就是指dev、prod这2个。
当然你也可以用命令行启动的时候带上参数：

```
1 java -jar xxx.jar --spring.profiles.active=dev
```

我给不同的环境添加不同的端口属性server.port，然后根据指定不同的spring.profiles.active来切换使用。各位可以自己试试。这里就不贴代码了。

除了可以用profile的配置文件来分区配置我们的环境变量，在代码里，我们还可以直接用@Profile注解来进行配置，例如数据库配置，这里我们先定义一个接口

igure(): }

查看 18 条评论

分别定义俩个实现类来实现它

```
/**
  * 测试数据库
  */
@Component
@Profile("testdb")
public class TestDBConnector implements DBConnector {
    @Override
    public void configure() {
        System.out.println("testdb");
    }
}
/**
 * 生产数据库
 */
@Component
@Profile("devdb")
public class DevDBConnector implements DBConnector {
    @Override
    public void configure() {
        System.out.println("devdb");
    }
}
```

通过在配置文件激活具体使用哪个实现类

```
spring.profiles.active=testdb
```

然后就可以这么用了

```
@RestController
@RequestMapping("/task")
public class TaskController {

    @Autowired DBConnector connector ;

    @RequestMapping(value = {"/",""})
    public String hellTask(){

        connector.configure(); //最终打印testdb
        return "hello task !! myage is " + myage;
    }
}
```

除了spring.profiles.active来激活一个或者多个profile之外，还可以用spring.profiles.include来叠加profile

```
spring.profiles.active: testdb
spring.profiles.include: proddb,prodmq
```

总结

这次对Spring Boot中application.properties配置文件做的整理总结希望对大家有所帮助，最后贴上Spring Boot中常用的配置属性，需要的时候可打开查找。

(￣▽￣)↗Spring Boot干货系列：常用属性汇总

想要查看更多Spring Boot干货教程,可前往：Spring Boot干货系列总纲

参考

Spring Boot属性配置文件详解
spring boot 使用profile来分区配置
Spring Boot实战
JavaEE开发的颠覆者Spring Boot实战

源码下载

(￣▽￣)↗[相关示例完整代码]

一直觉得自己写的不是技术，而是情怀，一篇篇文章是自己这一路走来的痕迹。靠专业技能的成功是最具可复制性的，希望我的这条路能让你少走弯路，希望我能帮你抹去知识的蒙尘，希望我能帮你理清知识的脉络，希望未来技术之巅上有你也有我,希望大爷你看完打赏点零花钱给我。

订阅博主微信公众号：嘟爷java超神学堂（javaLearn）三大好处：

- 获取最新博主博客更新信息，首发公众号
- 获取大量视频，电子书，精品破解软件资源
- 可以跟博主聊天，欢迎程序媛妹妹来撩我

博主最近发起了《嘟爷电子书互惠组》计划，里面包含了《精通Spring4.X企业应用开发实战》相关书籍在内的至少227本Java相关的电子书，也有博主花钱买的电子书。可谓新手必备之物，详情可前往书单末尾查看: Java后端2017书单推荐